



**Szkoła Główna
Handlowa
w Warszawie**

Kolegium Analiz Ekonomicznych

**Podyplomowe Studia
Inżynieria danych – Big Data**

Edycja: 14

**Praca dyplomowa
„Churn data analysis on the e-commerce company example”**

Autor: mgr inż. Linda Romańska
Promotor: dr Dominik Deja

nr albumu: lr113683

Warszawa 2022

Abstract

The thesis „Churn data analysis on the e-commerce company example” was made on the data set “Ecommerce Customer Churn Analysis and Prediction” found on Kaggle (<https://www.kaggle.com/datasets/ankitverma2010/ecommerce-customer-churn-analysis-and-prediction>, 2022). The research problem was to predict which customers would churn. It is a classification problem very typical for e-commerce sector. The classification methods used in prediction are Decision Tree, Bagging, Random Forest, Logistic Regression, Linear Discriminant Analysis, XGBoost, CatBoost and Neural Network. The algorithm that gave best results was XGBoost (accuracy 0.97, F1 0.89). After Hyper Parameter tuning aiming to increase recall score, the accuracy skyrocketed for 0.98 and F1 score for 0.93. It was determined that features that have biggest impact on prediction are Tenure and Complain.

Table of contents

Abstract.....	2
Table of contents	3
List of figures	6
List of tables.....	7
Research objectives	8
1. Introduction.....	9
1.1. Churn	9
1.1.1. The definition.....	9
1.1.2. Importancy	9
1.1.3. Churn metrics.....	9
1.2. Methodology	10
1.2.1. Decision tree	10
1.2.2. Bagging.....	10
1.2.3. Random forest.....	10
1.2.4. Logistic regression	10
1.2.5. Linear Discriminant Analysis	11
1.2.6. XGBoost	11
1.2.7. Neural network.....	11
1.2.8. CatBoost	11
1.3. Libraries.....	12
1.3.1. NumPy	12
1.3.2. Pandas.....	12
1.3.3. Scikit-learn (Sklern)	12
1.3.4. Matplotlib	12
1.3.5. Seaborn	12

1.3.6.	XGBoost	13
1.3.7.	CatBoost	13
1.4.	Model selection metrics	14
1.4.1.	Accuracy	14
1.4.2.	Precision	14
1.4.3.	Recall	14
1.4.4.	F1	15
1.4.5.	ROC AUC	15
2.	The data analysis	16
2.1.	The dataset	16
2.2.	Data exploration and visualization	18
2.2.1.	Churn rate	18
2.2.2.	Distribution of tenure of customer in organization	18
2.2.3.	Preferred login device of customer	19
2.2.4.	City tier of the customer	19
2.2.5.	Distribution of distance between warehouse and homes of customers	20
2.2.6.	Preferred payment method of customer	20
2.2.7.	Gender of customers	21
2.2.8.	Number of hours spend on mobile application or website	21
2.2.9.	Total number of devices registered for a particular customer	22
2.2.10.	Preferred order category of customer in last month	22
2.2.11.	Satisfaction score of customer on service	23
2.2.12.	Marital status of customer	23
2.2.13.	Total number of added addresses on particular customer	24
2.2.14.	Any complaint raised in last month	24
2.2.15.	Order amount increase from last year	25
2.2.16.	Total number of coupons used during last month	25

2.2.17.	Total number of orders placed during last month	26
2.2.18.	Days since last order.....	26
2.2.19.	Average cashback in last month.....	27
2.3.	Preprocessing	28
2.3.1.	Correlation map.....	28
2.3.2.	Data reduction.....	30
2.3.3.	Cleaning.....	30
2.3.4.	One hot encoding	33
2.3.5.	Scaling	34
2.3.6.	Dataset split.....	36
3.	Modelling.....	37
3.1.	Accuracy	37
3.2.	Other metrics.....	38
3.3.	Cross validation score.....	39
3.4.	ROC AUC score.....	39
3.5.	Confusion matrixes.....	40
3.6.	Feature importances	42
3.7.	Hyperparameter tuning	44
	Summary.....	45
	References.....	46
	Appendix 1 – Code.....	47
	Appendix 2 – Decision Tree	58

List of figures

Figure 1	18
Figure 2	18
Figure 3	19
Figure 4	19
Figure 5	20
Figure 6	20
Figure 7	21
Figure 8	21
Figure 9	22
Figure 10	22
Figure 11	23
Figure 12	23
Figure 13	24
Figure 14	24
Figure 15	25
Figure 16	25
Figure 17	26
Figure 18	26
Figure 19	27
Figure 20	28
Figure 21	32
Figure 22	33
Figure 23	43
Figure 24	43
Figure 25	58
Figure 26	59
Figure 27	60

List of tables

Table 1. Data frame before modifications	16
Table 2. Data characteristics rounded to two decimal places	17
Table 3. Data frame after CustomerID column removal	30
Table 4. Missing values	31
Table 5 Data frame with categorical values changed to dummies.....	34
Table 6. Numerical features after scaling	35
Table 7. Characteristics of numerical features.....	35
Table 8. Accuracy results.....	37
Table 9. Test metrics	38
Table 10. Cross validation score	39
Table 11. Roc auc score.....	39
Table 12. Confusion matrixes	40
Table 13. The most important features.....	42
Table 14. The parameters used for tuning	44
Table 15. The best parameters chosen.....	44
Table 16. Confusion matrix after tuning.....	44
Table 17. Libraries	47
Table 18. Countplot.....	49
Table 19. Displot.....	49
Table 20. Barh plot.....	49
Table 21. Correlation map	49
Table 22. Boxplot.....	50
Table 23. Data split.....	50
Table 24. Modelling	50
Table 25. Classification reports.....	53
Table 26. Confusion matrixes	53
Table 27. Importances	55
Table 28. Hyperparameter tuning.....	56

Research objectives

The aim of this thesis is to analyse the data about the churn from E-commerce company. I wanted to find out which characteristics the churning customers have, and which model would be the best to analyse the churn phenomenon.

After the introduction, there is a chapter about data exploration and visualization. Next part is about preprocessing of the data. The last one contains comparison of various models (Decision Tree, Bagging, Random Forest, Logistic Regression, Linear Discriminant Analysis, XGBoost, Neural Network). At the end there is a conclusion and overall summary.

1. Introduction

The introduction contains parts about methodology of the models used, the libraries, model selection metrics and the theory about the churn.

1.1. Churn

This chapter is about the churn phenomenon.

1.1.1. The definition

The definition of churn from Cambridge Academic Content dictionary is the situation in which customers stop buying the products or services of a particular company, especially to buy them from the competitor (<https://dictionary.cambridge.org/pl/dictionary/english/churn> , 2022).

1.1.2. Importancy

Churn analysis allows the company to evaluate strengths and weaknesses in the existing processes, while improving the ability to deal with, and prevent, future churn (<https://www.paddle.com/blog/churn-analysis>, 2022).

1.1.3. Churn metrics

The simplest formula to calculate churn rate is: Number of customers cancelling their subscription/stopping being a customer per time interval, divided by the number of customers at the beginning of that interval (<https://www.paddle.com/blog/churn-analysis>, 2022):

$$\text{Churn rate} = \frac{\text{number of churned customers}}{\text{total number of customers}}$$

But churn can be tricky to count as well. For example, for subscription there are different levels of churn: the customer can either cancel their account or downgrade their subscription. Moreover, churn can be seasonal or constant. Certain types of customers may leave more often than others and certain subscription levels may hold customers better than others (<https://www.paddle.com/blog/churn-analysis>, 2022).

More accurate method involves taking an average churn rate for every day of the month and dividing it by the average number of customers during that same time period (<https://www.paddle.com/blog/churn-analysis>, 2022).

1.2. Methodology

This part is about the algorithms used in the thesis.

1.2.1. Decision tree

Decision trees can be used for classification and regression. It uses a tree-like model of decisions. In decision analysis, a decision tree can be made to visually and explicitly represent decisions and the process of decision making (<https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052>, 2022). Decision trees allow to present algorithms with conditional control statements. They include branches that represent decision-making steps that can lead to a favourable result (Provost i Fawcett, 2019).

1.2.2. Bagging

Bagging (Bootstrap Aggregation) is an ensemble machine learning algorithm that combines the predictions from many decision trees. This is done by taking multiple bootstrap samples from the training dataset and fitting a decision tree on each. The predictions from the decision trees are then combined to give a more robust and accurate prediction than a single decision tree (<https://machinelearningmastery.com/bagging-ensemble-with-python/>, 2022). Predictions are made for classification problems by taking the majority vote prediction for the classes from across the predictions made by the decision trees. The bagged decision trees are effective because each decision tree is fit on a little different training dataset, which allows each tree to have minor discrepancies and make slightly different skilful predictions.

1.2.3. Random forest

Random forest is a supervised learning algorithm. It is an extension of the bagging method as it utilizes both bagging and feature randomness to create an uncorrelated forest of decision trees (<https://builtin.com/data-science/random-forest-algorithm>, 2022). This means that at each split of the tree, the model considers only a small subset of features rather than all the features of the model.

1.2.4. Logistic regression

Logistic regression is basically a supervised classification algorithm. In regression analysis, logistic regression (logit regression) is estimating the parameters of a logistic model (the coefficients in the linear combination. The independent variable in a regression design can be nominal, ordinal, interval, or quotient (<https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc>, 2022). In a classification problem, the target variable (or output), y , can take only discrete values for a given set of features (or inputs), X .

1.2.5. Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a generalization of Fisher's linear discriminant, a supervised method which is used in statistics to find a linear combination of features that characterizes or separates two or more classes of objects or events. The resulting combination may be used as a linear classifier or for dimensionality reduction before later classification (<https://www.mltut.com/linear-discriminant-analysis-python-complete-and-easy-guide/>, 2022).

1.2.6. XGBoost

XGBoost stands for Extreme Gradient Boosting. XGBoost is a tree-based ensemble machine learning algorithm which is a scalable machine learning system for tree boosting (<https://www.kdnuggets.com/2020/12/xgboost-what-when.html>, 2022). It uses accurate approximations to find the best tree model as it makes use of cache access patterns, data compression and sharding which are essential elements for building a scalable end-to-end system for tree boosting.

1.2.7. Neural network

Artificial neural network (ANN) is a machine learning technique trying to mimic human brain. ANN's model consists of two or more layers containing artificial neurons – small computing units processing input and passing output to neurons in other layers. Last layer, also known as output layer, is responsible for creating the final outcome (Raschka i Mirjalili, 2019).

1.2.8. CatBoost

CatBoost is a machine learning algorithm that uses gradient boosting on decision trees. It provides a framework which helps solving for categorical features using a permutation driven alternative compared to the classical algorithm (<https://catboost.ai/>, 2022).

1.3. Libraries

This part contains the information about Python libraries which were used for the thesis. Table 17 in Appendix1 contains all commands for libraries import.

1.3.1. NumPy

NumPy is a highly performant open-source Python library created in 2005 by Travis Oliphant for working with arrays. NumPy stands for Numerical Python. It also has functions for working in domain of linear algebra, fourier transform, and matrices (https://www.w3schools.com/python/numpy/numpy_intro.asp, 2022).

1.3.2. Pandas

Pandas is a software library written for data manipulation and analysis built on top of the Python programming language (<https://pandas.pydata.org/>, 2022). It offers data structures and operations for manipulating numerical tables and time series. It was open sourced in 2009. Python with pandas is in use in a wide variety of academic and commercial domains, including Finance, Neuroscience, Economics, Statistics, Advertising, Web Analytics etc.

1.3.3. Scikit-learn (Sklearn)

Scikit-learn (Sklearn) is the most robust and useful library for machine learning in Python (https://www.tutorialspoint.com/scikit_learn/scikit_learn_introduction.htm, 2022). This library, which is mostly written in Python, is built on top of NumPy, SciPy and Matplotlib. It provides a selection of efficient tools for machine learning and statistical modelling including regression, classification, clustering and dimensionality reduction. It has a very convenient interface in Python.

1.3.4. Matplotlib

Matplotlib is a comprehensive library for creating static, animated or interactive visualizations in Python (<https://matplotlib.org/>, 2022). The plots are in good quality which can be exported to many file formats. JupyterLab and many other Graphical User Interfaces have them built in.

1.3.5. Seaborn

Seaborn is a library for making statistical graphics in Python (<https://seaborn.pydata.org/introduction.html>, 2022). It builds upon Matplotlib and integrates closely with pandas data structures. Its plotting functions operate on dataframes and arrays containing whole datasets. Seaborn helps exploring and understanding the data by performing the necessary semantic mapping and statistical aggregation to produce informative plots.

1.3.6. XGBoost

XGBoost (eXtreme Gradient Boosting) is an open-source optimized distributed gradient boosting library designed to be highly efficient, flexible and portable (<https://xgboost.readthedocs.io/en/stable/>, 2022). XGBoost provides a regularizing gradient boosting framework for C++, Java, Python, R, Julia, Perl and Scala. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way.

1.3.7. CatBoost

CatBoost is an open-source software library developed by Yandex (<https://catboost.ai/>, 2022). It provides a gradient boosting framework. The source code is licensed under Apache License and available on GitHub. It works in Python and R but models built using CatBoost can be used for predictions as well in C++, Java, C#, Rust, Core ML, ONNX, and PMML.

1.4. Model selection metrics

This chapter is about model selection metrics used to compare the models. They are using the data from the confusion matrix (Raschka i Mirjalili, 2019):

		Predicted class	
		T	F
True class	T	True positive (TP)	False negative (FN)
	N	False positive (FP)	True negative (TN)

1.4.1. Accuracy

The accuracy score is the fraction of true positives and true negatives over the total number of assigned labels (<https://towardsdatascience.com/accuracy-recall-precision-f-score-specificity-which-to-optimize-on-867d3f11124>, 2022) and is calculated as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Where TP = True positive; FP = False positive; TN = True negative; FN = False negative.

1.4.2. Precision

Precision (positive predictive value) tells us how many of the values we predicted to be in a certain class are actually in that class (<https://towardsdatascience.com/accuracy-recall-precision-f-score-specificity-which-to-optimize-on-867d3f11124>, 2022). Essentially, this tells us how we performed in terms of false positives. It is calculated as:

$$Precision = \frac{TP}{TP + FP}$$

Where TP = True positive; FP = False positive.

1.4.3. Recall

Recall (sensitivity/true positive rate) tells us how many of the values in each class were given the correct label, thus telling us how it performed relative to false negatives (<https://towardsdatascience.com/accuracy-recall-precision-f-score-specificity-which-to-optimize-on-867d3f11124>, 2022). It is calculated as:

$$Recall = \frac{TP}{TP + FN}$$

Where TP = True positive; FP = False positive; FN = False negative.

1.4.4. F1

F1 (F-score) is the harmonic mean of precision and recall. It prevents overestimating the performance of the model in cases where one parameter is high and the other low (<https://towardsdatascience.com/accuracy-recall-precision-f-score-specificity-which-to-optimize-on-867d3f11124>, 2022). It is calculated as:

$$F1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

1.4.5. ROC AUC

Area under Curve (AUC) or Receiver operating characteristic (ROC) curve is used to evaluate and compare the performance of binary classification model. ROC is a probability curve and AUC represents the degree or measure of separability. It tells us how well the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1 (<https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5>, 2022).

ROC can be drawn as chart with FPR on x axis and recall (sensitivity, TPR) on y axis. AUC is an area under this curve.

$$\text{Specificity} = \frac{TN}{TN + FP}$$

Where FP = False positive; TN = True negative.

$$FPR = 1 - \text{Specificity} = \frac{FP}{TN + FP}$$

Where FP = False positive; TN = True negative.

2. The data analysis

2.1. The dataset

The dataset belongs to a leading online E-Commerce company. An on-line retail (E-commerce) company wants to know the customers who are going to churn, so accordingly they can approach this group of customers and stop them leaving the organization.

File E-Commerce-Dataset was downloaded from Kaggle database (<https://www.kaggle.com/datasets/ankitverma2010/ecommerce-customer-churn-analysis-and-prediction>, 2022). The second excel sheet was then read into data frame in Jupiter Notebook. Table 1 contains information about all columns and Table 2 contains data characteristics.

Table 1. Data frame before modifications

#	Column	Non-Null Count	Dtype
---	-----	-----	-----
0	CustomerID	5630 non-null	int64
1	Churn	5630 non-null	int64
2	Tenure	5366 non-null	float64
3	PreferredLoginDevice	5630 non-null	object
4	CityTier	5630 non-null	int64
5	WarehouseToHome	5379 non-null	float64
6	PreferredPaymentMode	5630 non-null	object
7	Gender	5630 non-null	object
8	HourSpendOnApp	5375 non-null	float64
9	NumberOfDeviceRegistered	5630 non-null	int64
10	PreferedOrderCat	5630 non-null	object
11	SatisfactionScore	5630 non-null	int64
12	MaritalStatus	5630 non-null	object
13	NumberOfAddress	5630 non-null	int64
14	Complain	5630 non-null	int64
15	OrderAmountHikeFromlastYear	5365 non-null	float64
16	CouponUsed	5374 non-null	float64
17	OrderCount	5372 non-null	float64
18	DaySinceLastOrder	5323 non-null	float64
19	CashbackAmount	5630 non-null	float64

The data types are mainly correct. However, ‘Number of hours spend on mobile application or website’ and ‘Coupon used’ are two features which have only integer numbers, they are in float64 instead of int64 format. For that’s reason, barh plots are not possible to make and distribution plots will present characteristic peaks.

Table 2. Data characteristics rounded to two decimal places

	Customer ID	Churn	Tenure	CityTier	Warehouse ToHome	Hour Spend OnApp	NumberOf Device Registered	Satisfaction Score	Number Of Address	Complain	Order Amount Hike FromlastYear	Coupon Used	Order Count	DaySince LastOrder	Cashback Amount
count	5630.00	5630.00	5366.00	5630.00	5379.00	5375.00	5630.00	5630.00	5630.00	5630.00	5365.00	5374.00	5372.00	5323.00	5630.00
mean	52815.50	0.17	10.19	1.65	15.64	2.93	3.69	3.07	4.21	0.28	15.71	1.75	3.01	4.54	177.22
std	1625.39	0.37	8.56	0.92	8.53	0.72	1.02	1.38	2.58	0.45	3.68	1.89	2.94	3.65	49.21
min	50001.00	0.00	0.00	1.00	5.00	0.00	1.00	1.00	1.00	0.00	11.00	0.00	1.00	0.00	0.00
25%	51408.25	0.00	2.00	1.00	9.00	2.00	3.00	2.00	2.00	0.00	13.00	1.00	1.00	2.00	145.77
50%	52815.50	0.00	9.00	1.00	14.00	3.00	4.00	3.00	3.00	0.00	15.00	1.00	2.00	3.00	163.28
75%	54222.75	0.00	16.00	3.00	20.00	3.00	4.00	4.00	6.00	1.00	18.00	2.00	3.00	7.00	196.39
max	55630.00	1.00	61.00	3.00	127.00	5.00	6.00	5.00	22.00	1.00	26.00	16.00	16.00	46.00	324.99

2.2. Data exploration and visualization

For all features the plots were made, either countplot (Table 18, Appendix 1), displot (Table 19, Appendix 1) or barh plot (Table 20, Appendix 1).

2.2.1. Churn rate

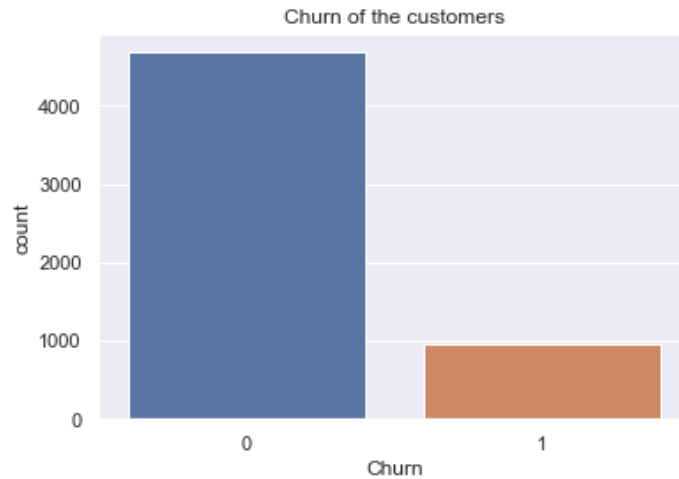


Figure 1

There is 17% churn rate in analysed company (Figure 1). Churn varies by industry and company, but the average churn rate for eCommerce ranges from 3% to nearly 13% (<https://www.winback.chat/blog/how-to-reduce-churn-in-ecommerce>, 2022) so the churn in our example is high.

2.2.2. Distribution of tenure of customer in organization



Figure 2

Most of the customers are new to organization (Figure 2) But there are outliers.

2.2.3. Preferred login device of customer

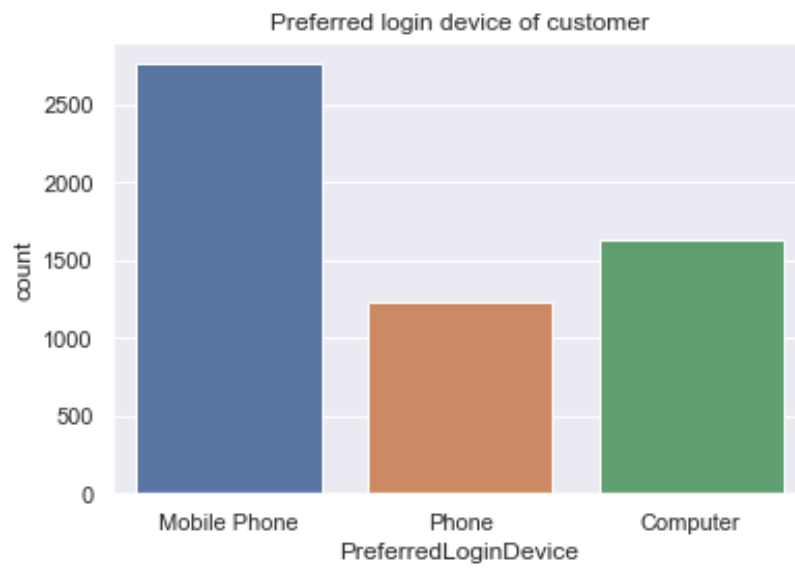


Figure 3

Most of the customers prefer to communicate via mobile phone (Figure 3).

2.2.4. City tier of the customer



Figure 4

Most of the customers live in city tier '1' (Figure 4).

2.2.5. Distribution of distance between warehouse and homes of customers



Figure 5

Most of the customers live close to the warehouse so the time of the delivery should not be an issue for the customers. But there are outliers (Figure 5).

2.2.6. Preferred payment method of customer

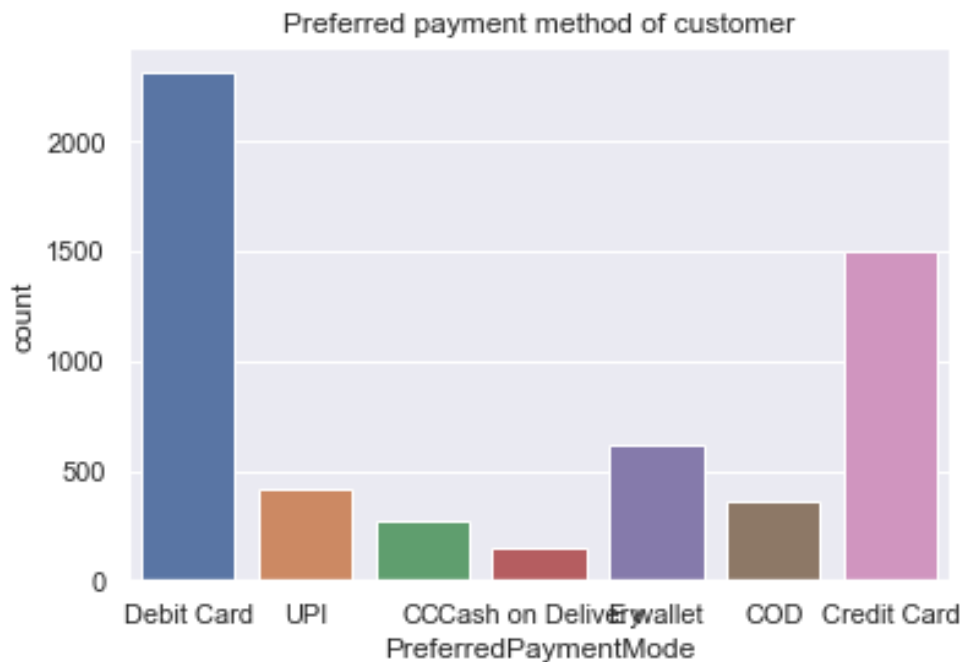


Figure 6

Most of the transactions are done by debit and credit cards (Figure 6).

2.2.7. Gender of customers



Figure 7

There are more male customers in the organization (Figure 7).

2.2.8. Number of hours spend on mobile application or website

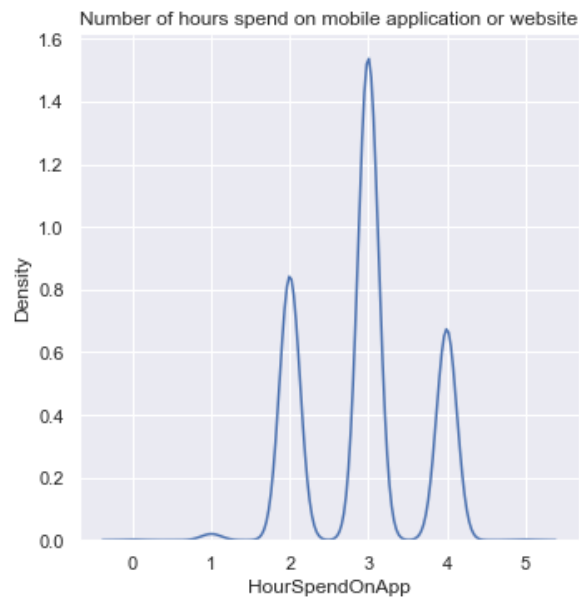


Figure 8

For the column HourSpendOnApp the mode is 3 hours (Figure 8).

2.2.9. Total number of devices registered for a particular customer

Total number of deceives registered on particular customer

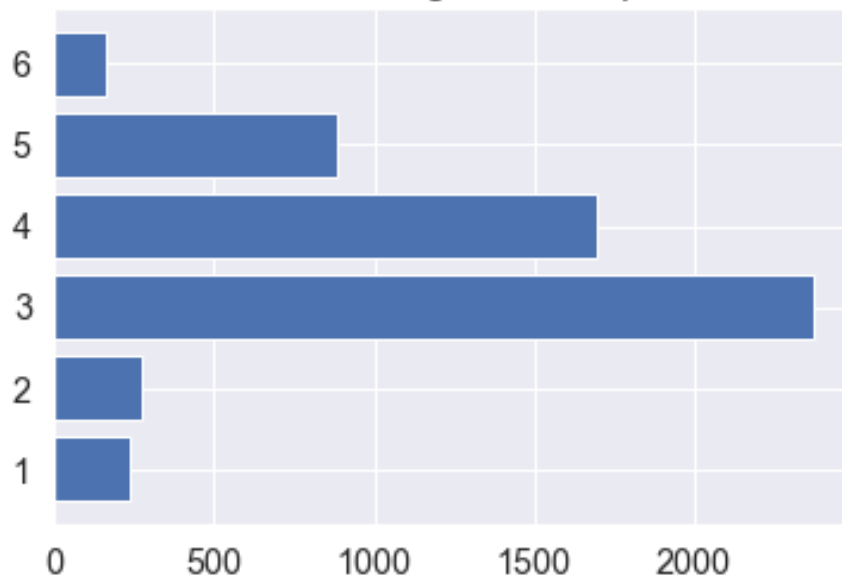


Figure 9

Customers usually use many devices (Figure 9).

2.2.10. Preferred order category of customer in last month

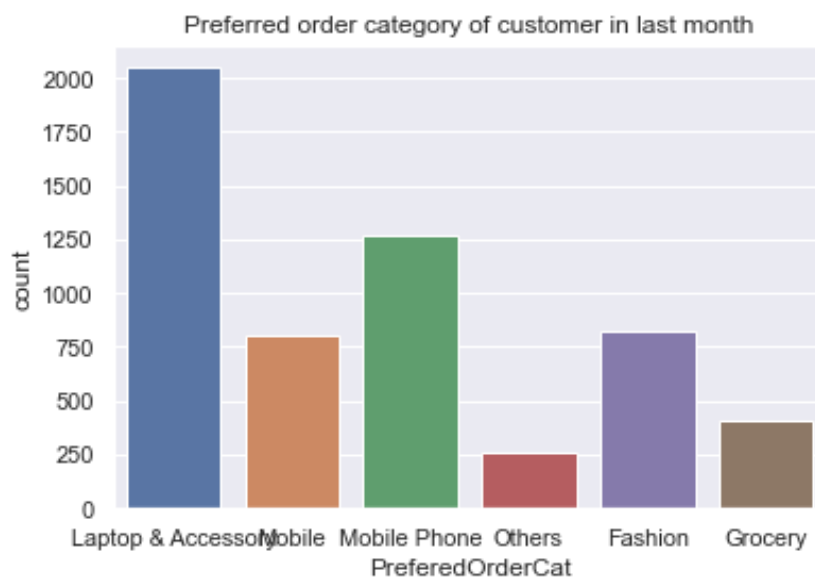


Figure 10

The most popular categories of orders in last month are laptops & accessory and mobile phones (Figure 10).

2.2.11. Satisfaction score of customer on service



Figure 11

There are quite many customers who are not satisfied with the service (Figure 11).

2.2.12. Marital status of customer

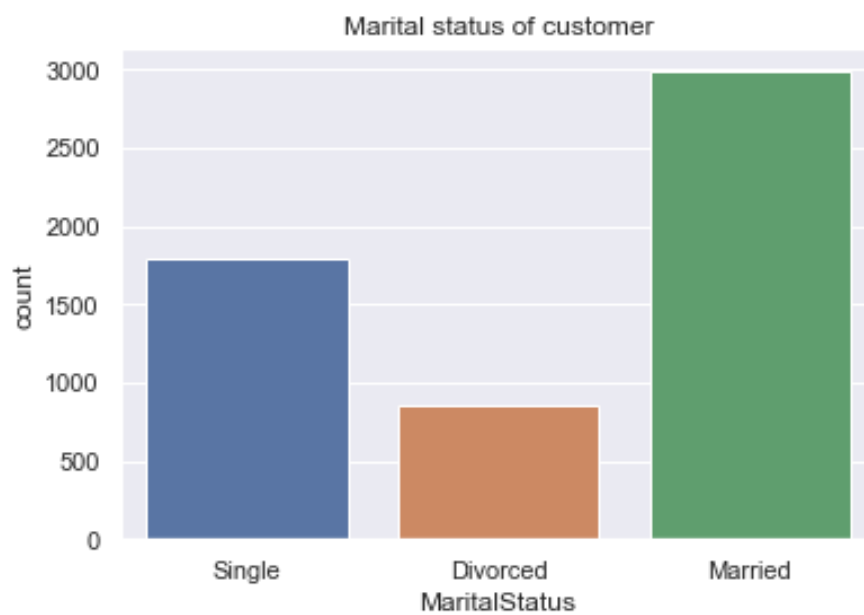


Figure 12

Most of the customers are married (Figure 12).

2.2.13. Total number of added addresses on particular customer

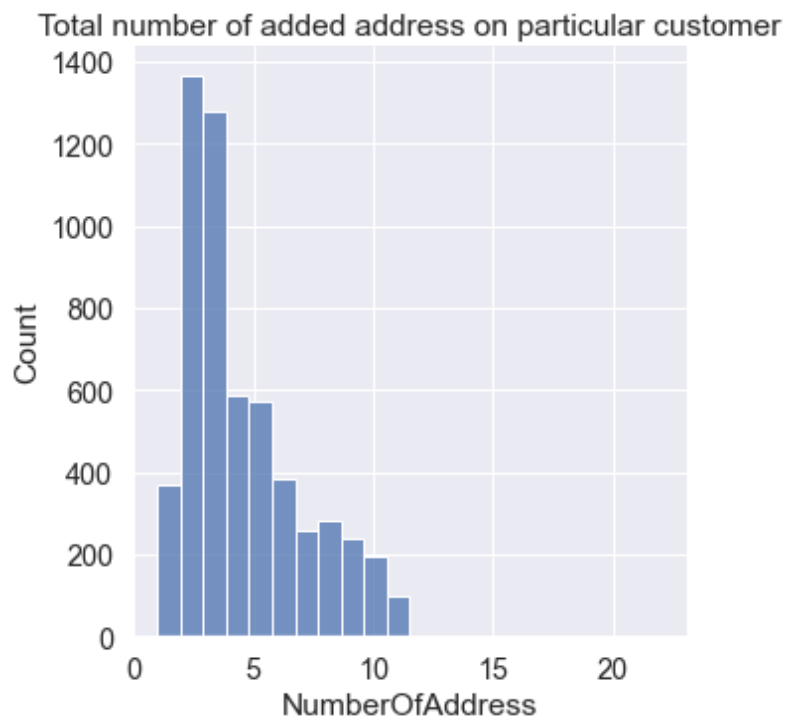


Figure 13

Most of the customers have two or three addresses (Figure 13).

2.2.14. Any complaint raised in last month

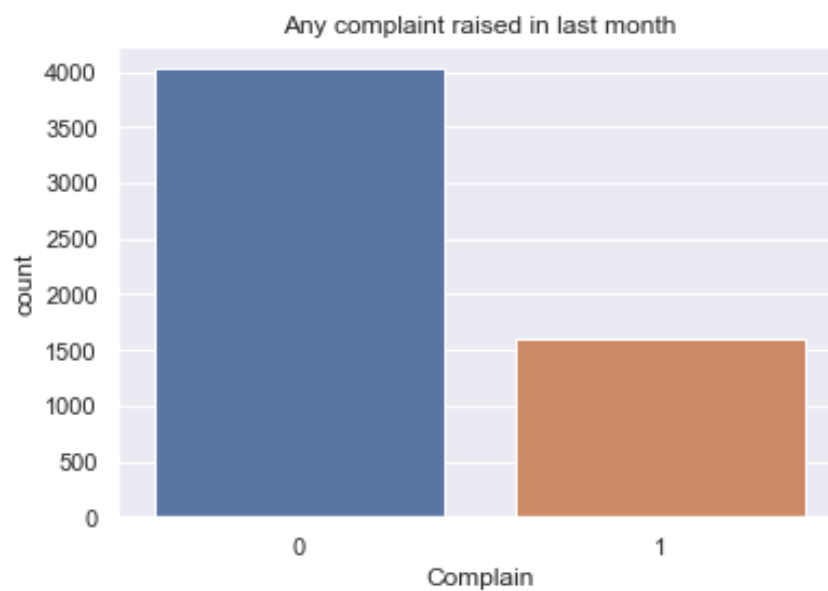


Figure 14

28% of the customers raised a complaint during last month (Figure 14).

2.2.15. Order amount increase from last year

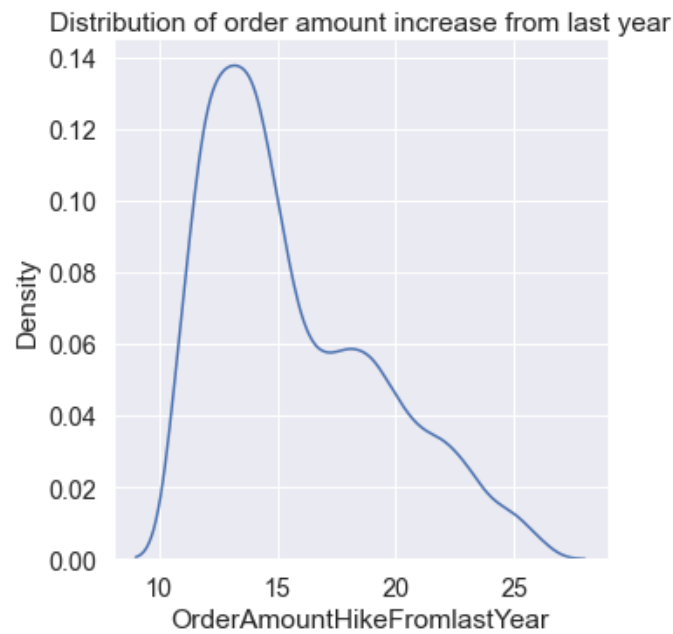


Figure 15

The mean of order amount hike from last year is 15,7, however the median is lower as the right tail is long (Figure 15).

2.2.16. Total number of coupons used during last month



Figure 16

The mean of total number of coupons used during last month is 1,75 but the median is higher because of long right tail (Figure 16). All values are integers, so the distribution plot has waves.

2.2.17. Total number of orders placed during last month



Figure 17

The mean of orders placed during last month is 3 but there are customers who placed more than 15 orders (Figure 17).

2.2.18. Days since last order

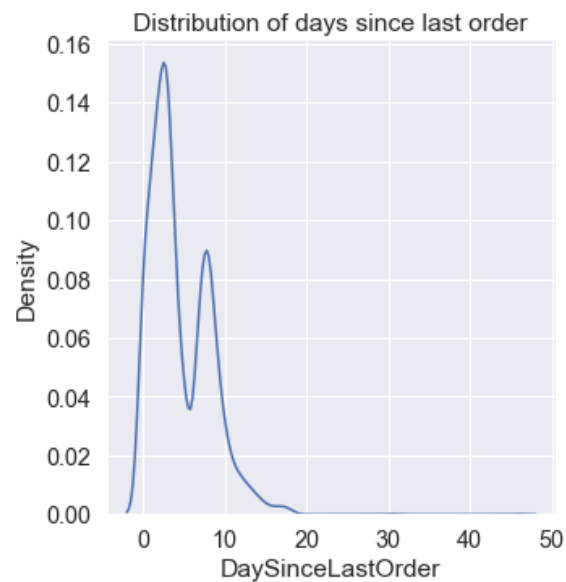


Figure 18

The mean of days since last order is 4.54 but there are some outliers (Figure 18).

2.2.19. Average cashback in last month

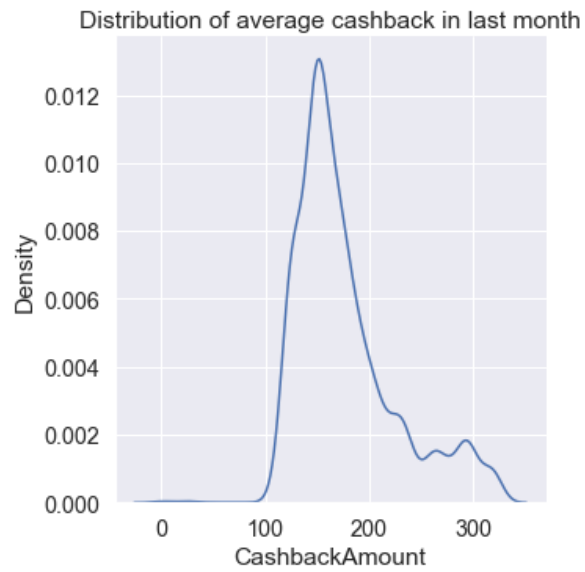


Figure 19

The average cashback of customer last month was below 200 but there are some customers who order usually above 300 (Figure 19).

2.3. Preprocessing

The data before modelling was checked for correlation. The data reduction, data cleaning, one hot encoding and scaling were done before dataset split.

2.3.1. Correlation map

The correlation map is showed on Figure 20 (the code is displayed in Table 21, Appendix 1).

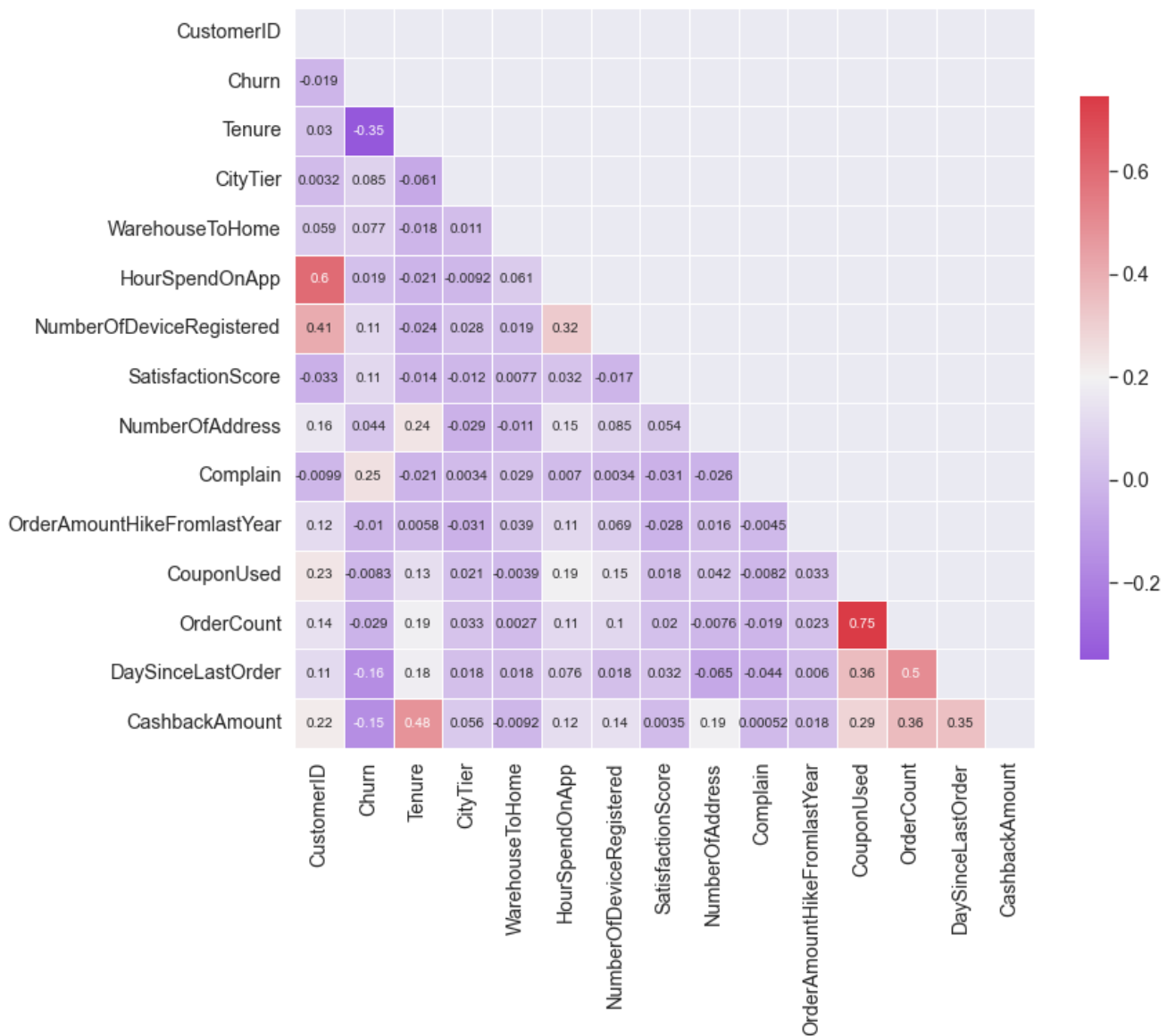


Figure 20

Churn has a positive correlation with CityTier, WarehouseToHome, NumberOfDeviceRegistered, SatisfactionScore and Complain. Churn has a negative correlation with Tenure, DaySinceLastOrder, and CashbackAmount.

Churn has a very low (no) correlation with HourSpendOnApp, NumberOfAddress, OrderAmountHikeFromlastYear, CouponUsed and OrderCount.

The strong correlated values are OrderCount and CouponUsed. Maybe the customers make more orders if they have coupons expiring.

The HourSpendOnApp looks as be correlated with CustomerID but it should be a coincidence. CustomerID column will be cut anyway.

2.3.2. Data reduction

The CustomerID column was cut with the drop command. The remaining data are presented in Table 3.

Table 3. Data frame after CustomerID column removal

Data columns (total 19 columns):			
#	Column	Non-Null Count	Dtype
---	-----	-----	-----
0	Churn	5630 non-null	int64
1	Tenure	5630 non-null	float64
2	PreferredLoginDevice	5630 non-null	object
3	CityTier	5630 non-null	int64
4	WarehouseToHome	5630 non-null	float64
5	PreferredPaymentMode	5630 non-null	object
6	Gender	5630 non-null	object
7	HourSpendOnApp	5630 non-null	float64
8	NumberOfDeviceRegistered	5630 non-null	int64
9	PreferedOrderCat	5630 non-null	object
10	SatisfactionScore	5630 non-null	int64
11	MaritalStatus	5630 non-null	object
12	NumberOfAddress	5630 non-null	int64
13	Complain	5630 non-null	int64
14	OrderAmountHikeFromlastYear	5630 non-null	float64
15	CouponUsed	5630 non-null	float64
16	OrderCount	5630 non-null	float64
17	DaySinceLastOrder	5630 non-null	float64
18	CashbackAmount	5630 non-null	float64
dtypes: float64(8), int64(6), object(5)			

2.3.3. Cleaning

The data was checked for missing values with isnull command (missingno package).

Table 4. Missing values

Churn	0
Tenure	264
PreferredLoginDevice	0
CityTier	0
WarehouseToHome	251
PreferredPaymentMode	0
Gender	0
HourSpendOnApp	255
NumberOfDeviceRegistered	0
PreferedOrderCat	0
SatisfactionScore	0
MaritalStatus	0
NumberOfAddress	0
Complain	0
OrderAmountHikeFromlastYear	265
CouponUsed	256
OrderCount	258
DaySinceLastOrder	307
CashbackAmount	0

The missing values are presented in Table 4. There are nulls in 7 columns: Tenure, WarehouseToHome, HourSpendOnApp, OrderAmountHikeFromlastYear, CouponUsed, OrderCount and DaySinceLastOrder. They are all float64 type columns. The only type float64 column which has no nulls is CashbackAmount one.

Afterwards, there was made a boxplot to study outliers (the code is displayed in Table 22, Appendix 1).

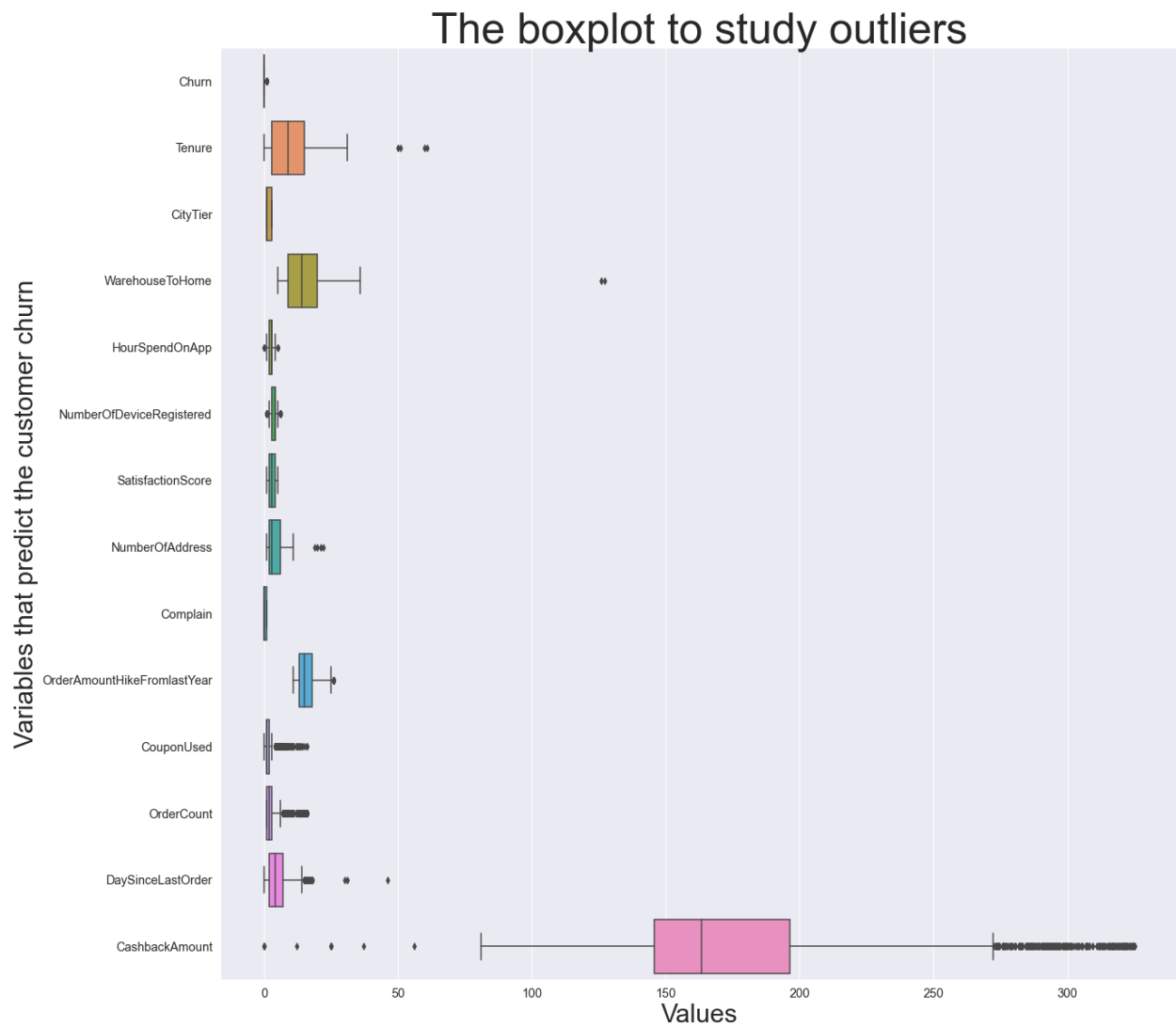


Figure 21

There are outliers especially for float64 columns (Figure 21). So, these outliers would be changed.

The subtraction between 3rd and 1st quartiles was calculated. When the distance between outlier and 1.5 times this subtraction was higher than the outlier value, it was changed for this subtraction.

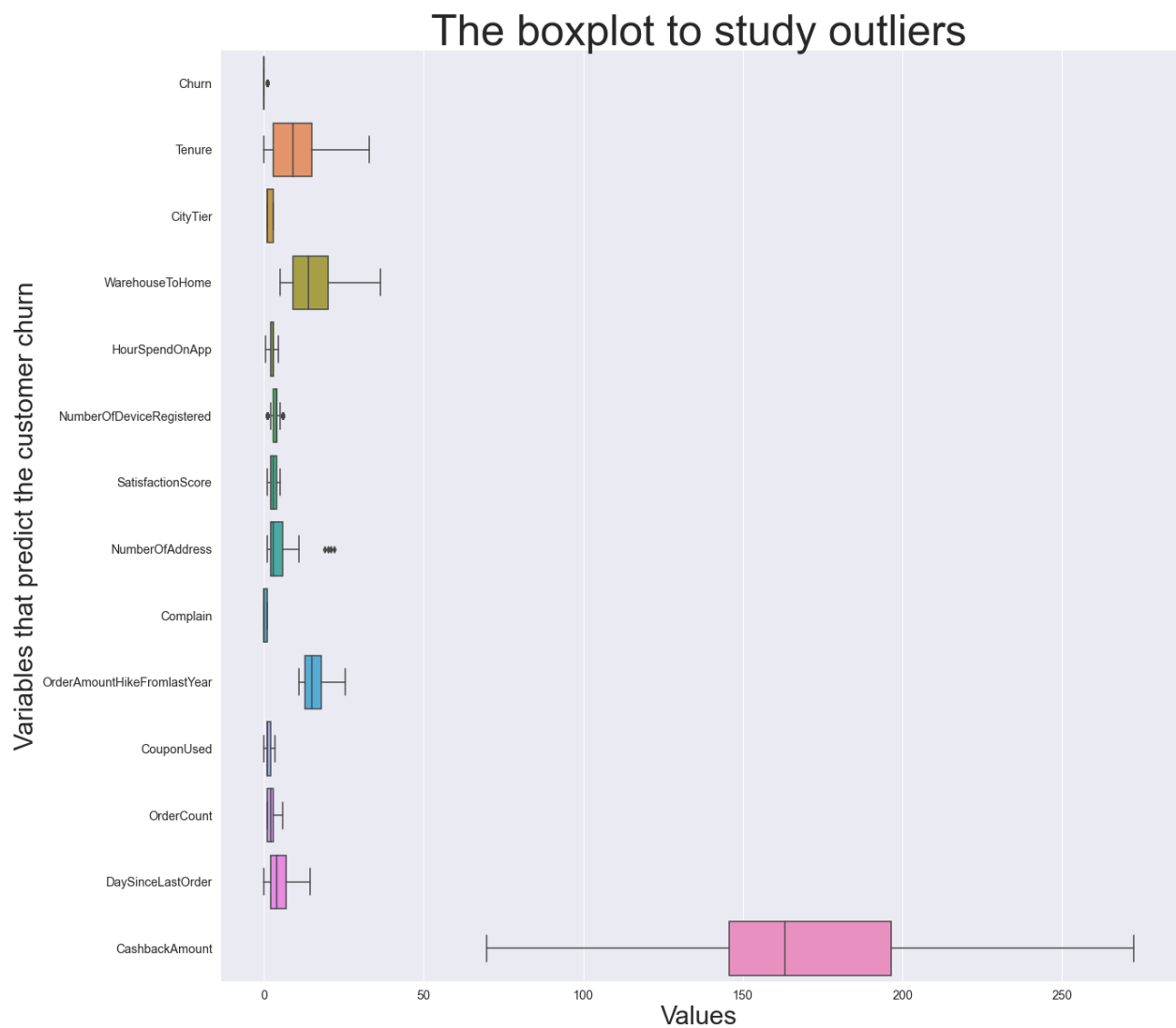


Figure 22

There are only outliers for NumberOfAddress left because this column is type int64 not float64 like these which were changed (Figure 22).

2.3.4. One hot encoding

The categorical values for making models must have been changed for dummies

Table 5 Data frame with categorical values changed to dummies

Data columns (total 30 columns):				
#	Column	Non-Null Count		Dtype
---	-----	-----		-----
0	Churn	5630	non-null	int64
1	Tenure	5630	non-null	float64
2	CityTier	5630	non-null	int64
3	WarehouseToHome	5630	non-null	float64
4	HourSpendOnApp	5630	non-null	float64
5	NumberOfDeviceRegistered	5630	non-null	int64
6	SatisfactionScore	5630	non-null	int64
7	NumberOfAddress	5630	non-null	int64
8	Complain	5630	non-null	int64
9	OrderAmountHikeFromlastYear	5630	non-null	float64
10	CouponUsed	5630	non-null	float64
11	OrderCount	5630	non-null	float64
12	DaySinceLastOrder	5630	non-null	float64
13	CashbackAmount	5630	non-null	float64
14	PreferredLoginDevice_Mobile Phone	5630	non-null	uint8
15	PreferredLoginDevice_Phone	5630	non-null	uint8
16	PreferredPaymentMode_COD	5630	non-null	uint8
17	PreferredPaymentMode_Cash on Delivery	5630	non-null	uint8
18	PreferredPaymentMode_Credit Card	5630	non-null	uint8
19	PreferredPaymentMode_Debit Card	5630	non-null	uint8
20	PreferredPaymentMode_E wallet	5630	non-null	uint8
21	PreferredPaymentMode_UPI	5630	non-null	uint8
22	Gender_Male	5630	non-null	uint8
23	PreferedOrderCat_Grocery	5630	non-null	uint8
24	PreferedOrderCat_Laptop & Accessory	5630	non-null	uint8
25	PreferedOrderCat_Mobile	5630	non-null	uint8
26	PreferedOrderCat_Mobile Phone	5630	non-null	uint8
27	PreferedOrderCat_Others	5630	non-null	uint8
28	MaritalStatus_Married	5630	non-null	uint8
29	MaritalStatus_Single	5630	non-null	uint8
dtypes: float64(8), int64(6), uint8(16)				

After this operation there are 11 more columns in the dataframe (Table 5).

2.3.5. Scaling

The best for scaling of numerical features (type int64 or float64) for classification problems is standardization. The sklearn StandardScaler was used. After using the scaler, the data frame was made from the array using DataFrame function from pandas. Of course, the Churn column was not standardized. Characteristics of numerical features before and after standardization are in the Table 6 and 7.

Table 7. Characteristics of numerical features

	count	mean	std	min	25%	50%	75%	max
Tenure	5630.0	10.173914	8.287624	0.00000	3.00	9.00	15.000000	33.000000
CityTier	5630.0	1.654707	0.915389	1.00000	1.00	1.00	3.000000	3.000000
WarehouseToHome	5630.0	15.607924	8.082533	5.00000	9.00	14.00	20.000000	36.500000
HourSpendOnApp	5630.0	2.931535	0.703682	0.50000	2.00	3.00	3.000000	4.500000
NumberOfDeviceRegistered	5630.0	3.688988	1.023999	1.00000	3.00	4.00	4.000000	6.000000
SatisfactionScore	5630.0	3.066785	1.380194	1.00000	2.00	3.00	4.000000	5.000000
NumberOfAddress	5630.0	4.214032	2.583586	1.00000	2.00	3.00	6.000000	22.000000
Complain	5630.0	0.284902	0.451408	0.00000	0.00	0.00	1.000000	1.000000
OrderAmountHikeFromlastYear	5630.0	15.704991	3.579711	11.00000	13.00	15.00	18.000000	25.500000
CouponUsed	5630.0	1.474558	1.068679	0.00000	1.00	1.00	2.000000	3.500000
OrderCount	5630.0	2.569295	1.715059	1.00000	1.00	2.00	3.008004	6.020011
DaySinceLastOrder	5630.0	4.513295	3.422020	0.00000	2.00	4.00	7.000000	14.500000
CashbackAmount	5630.0	175.324229	44.069817	69.83625	145.77	163.28	196.392500	272.32625

Table 6. Numerical features after scaling

	min	25%	50%	75%	max
0	-1.23	-0.87	-0.14	0.58	2.75
1	-0.72	-0.72	-0.72	1.47	1.47
2	-1.31	-0.82	-0.20	0.54	2.59
3	-3.46	-1.32	0.10	0.10	2.23
4	-2.63	-0.67	0.30	0.30	2.26
5	-1.50	-0.77	-0.05	0.68	1.40
6	-1.24	-0.86	-0.47	0.69	6.88
7	-0.63	-0.63	-0.63	1.58	1.58
8	-1.31	-0.76	-0.20	0.64	2.74
9	-1.38	-0.44	-0.44	0.49	1.90
10	-0.91	-0.92	-0.33	0.26	2.01
11	-1.32	-0.73	-0.15	0.73	2.91
12	-2.39	-0.67	-0.27	0.48	2.20

The encoded categorical columns were then concatenated (with `concat` function from `pandas`) with standardized numerical columns.

2.3.6. Dataset split

From the data frame after standardization, the two other frames were made: for dependent features (Churn) and independent features (all other columns beside Churn).

Afterwards, the dataset was split: 70% of data went for training and 30% went for testing. It was done using `train_test_split` from `sklearn.model.selection`. The code is displayed in Table 23, Appendix 1.

3. Modelling

Decision Tree, Bagging, Random Forest, Logistic Regression, Linear Discriminant Analysis, XGBoost, CatBoost and Neural Network were performed. The code is available in Table 24, Appendix 1.

The decision tree was plotted with `sklearn.tree.plot_tree` method (schema on Figure 25, Appendix 2), `sklearn.tree.export_graphviz` method (a piece on Figure 27, Appendix 2) and with `dtreeviz` package (a piece on Figure 26, Appendix 2).

3.1. Accuracy

The highest accuracy results have the models of XGBoost (0.97), Random Forest and Bagging (0.96) – Table 8. It means that XGBoost correctly indicated 97% of total observations.

Table 8. Accuracy results

	accuracy
Decision Tree	0.95
Bagging	0.96
Random Forest	0.96
Logistic Regression	0.89
Linear Discriminant Analysis	0.89
XGBoost	0.97
CatBoost	0.95
Neural Network	0.91

3.2. Other metrics

The other metrics are presented in Table 9 (the code is displayed in Table 25, Appendix 1). The best F1-score has XGBoost (92%). Recall for this classification model is 88% which means we can correctly identify 86% of the customers that will leave. The precision is 96% which means out of the predicted customers that will leave, 99% will actually leave. High precision rates have as well Random Forest, Bagging and CatBoost but they have lower recall rate. Recall rate is high for Decision Tree but it has pretty low precision (only 86%).

Table 9. Test metrics

	precision	recall	f1-score
Decision Tree	0.86	0.83	0.84
Bagging	0.93	0.81	0.87
Random Forest	0.95	0.79	0.86
Logistic Regression	0.77	0.51	0.62
Linear Discriminant Analysis	0.75	0.50	0.60
XGBoost	0.96	0.88	0.92
CatBoost	0.92	0.79	0.85
Neural Network	0.77	0.64	0.70

3.3. Cross validation score

Cross validation score (for F1) is next metric which is the highest for XGBoost method (86%). Good results have also CatBoost (81%) and Random Forest (80%) – Table 10. There is no “1” result so there is no overfitting.

Table 10. Cross validation score

	cross validation score
Decision Tree	0.79
Bagging	0.78
Random Forest	0.83
Logistic Regression	0.62
Linear Discriminant Analysis	0.60
XGBoost	0.86
CatBoost	0.81
Neural Network	0.70

3.4. ROC AUC score

ROC AUC score is the highest for XGBoost method (93%). It means that the model is close to perfect in distinguishing between classes. Good results have also Decision Tree and Bagging (both 90%) – Table 10.

Table 11. Roc auc score

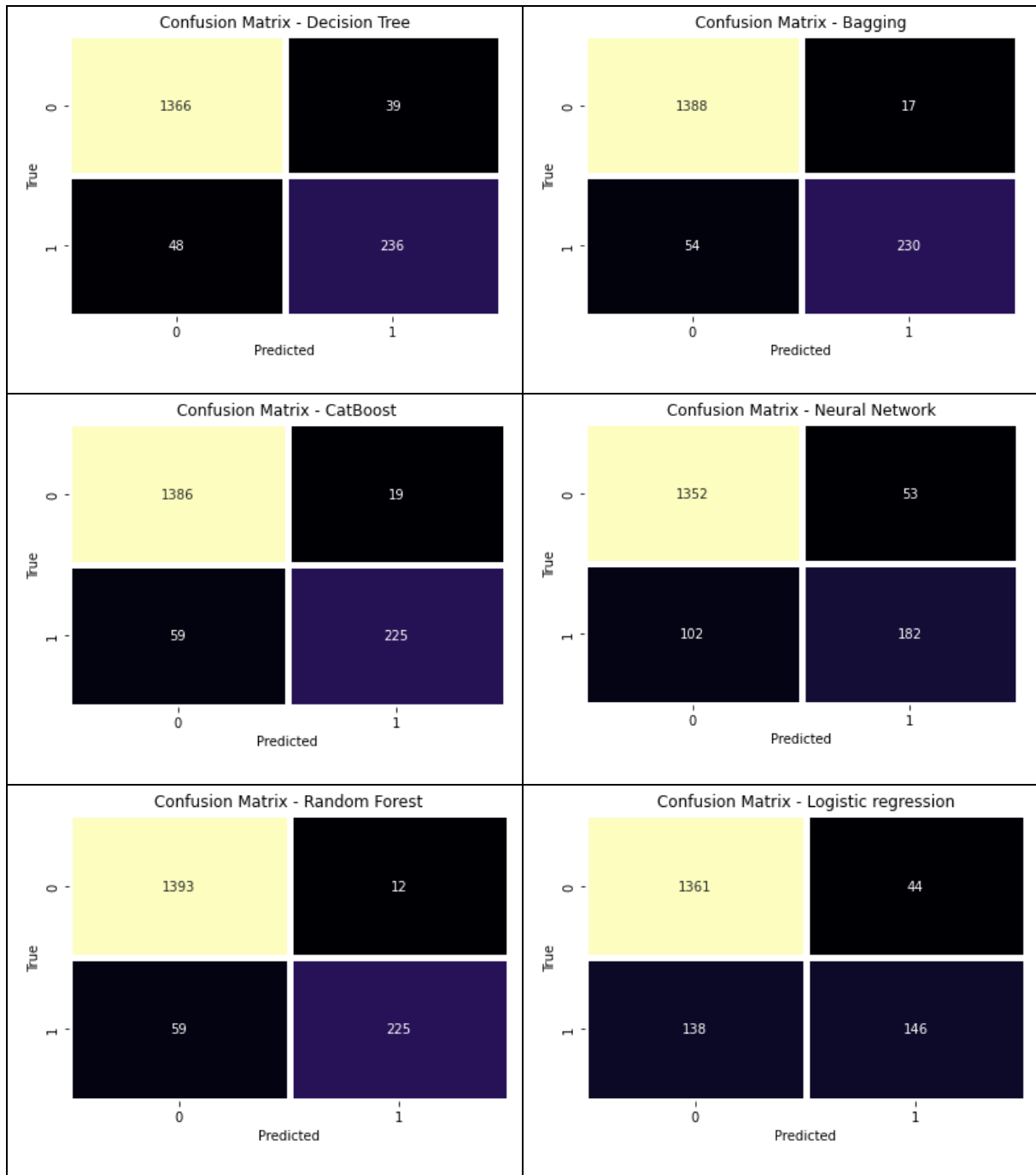
	ROC AUC score
Decision Tree	0.90
Bagging	0.90
Random Forest	0.89
Logistic Regression	0.74
Linear Discriminant Analysis	0.73
XGBoost	0.93
CatBoost	0.89
Neural Network	0.80

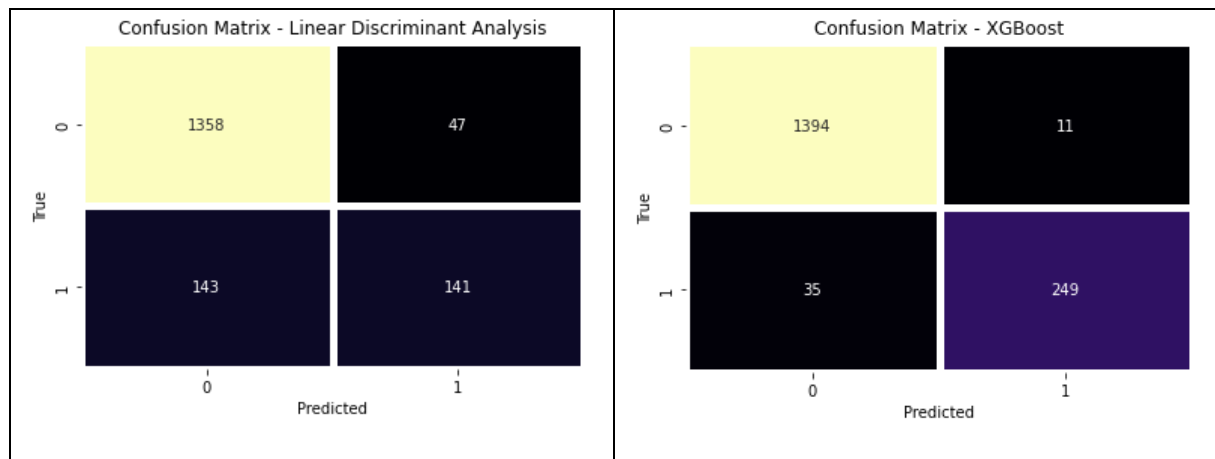
3.5. Confusion matrixes

Confusion matrixes from all models are in Table 12. The code is in Table 26, Appendix 1.

Comparing the models, the lowest sum of false positives and false negatives have XGBoost (11 + 35), Random Forest (12 + 59) and Bagging (17 + 54).

Table 12. Confusion matrixes





3.6. Feature importances

Table 13. The most important features

Feature importances - Decision Tree		
	feature	importance
16	Tenure	0.274
28	CashbackAmount	0.095
18	WarehouseToHome	0.083
22	NumberOfAddress	0.080
27	DaySinceLastOrder	0.070
23	Complain	0.065
21	SatisfactionScore	0.049
24	OrderAmountHikeFromlastYear	0.048
20	NumberOfDeviceRegistered	0.032
15	MaritalStatus_Single	0.022
17	CityTier	0.022
Feature importances - Random Forest		
	feature	importance
16	Tenure	0.221
28	CashbackAmount	0.097
18	WarehouseToHome	0.072
22	NumberOfAddress	0.063
23	Complain	0.062
24	OrderAmountHikeFromlastYear	0.058
27	DaySinceLastOrder	0.058
21	SatisfactionScore	0.050
20	NumberOfDeviceRegistered	0.038
25	CouponUsed	0.027
26	OrderCount	0.027
Feature importances - CatBoost		
	feature	importance
16	Tenure	18.754
21	SatisfactionScore	8.180
23	Complain	8.038
22	NumberOfAddress	7.702
28	CashbackAmount	7.671
18	WarehouseToHome	7.488
27	DaySinceLastOrder	5.352
24	OrderAmountHikeFromlastYear	5.209
10	PreferedOrderCat_Laptop & Accessory	4.955
17	CityTier	4.618
Feature importances - XGBoost		
	feature	importance
16	Tenure	0.122
23	Complain	0.082
10	PreferedOrderCat_Laptop & Accessory	0.058
6	PreferredPaymentMode_E wallet	0.051
22	NumberOfAddress	0.044
9	PreferedOrderCat_Grocery	0.044
15	MaritalStatus_Single	0.042
21	SatisfactionScore	0.037
27	DaySinceLastOrder	0.035
11	PreferedOrderCat_Mobile	0.034

Some models allow to check feature importances (Table 11, the code – Table 27, Appendix 1). The different features were important for different algorithms. But Tenure was important for all Decision Tree, Random Forest, CatBoost and XGBoost. The longer is the customer in the company the more likely he won't churn (Figure 23).

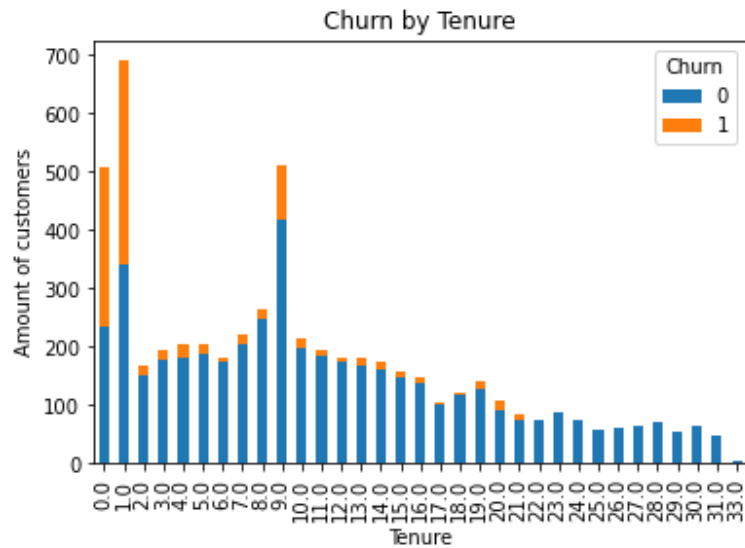


Figure 23

The other important factor was Complain. If the customer complains, he is more likely to churn (Figure24).

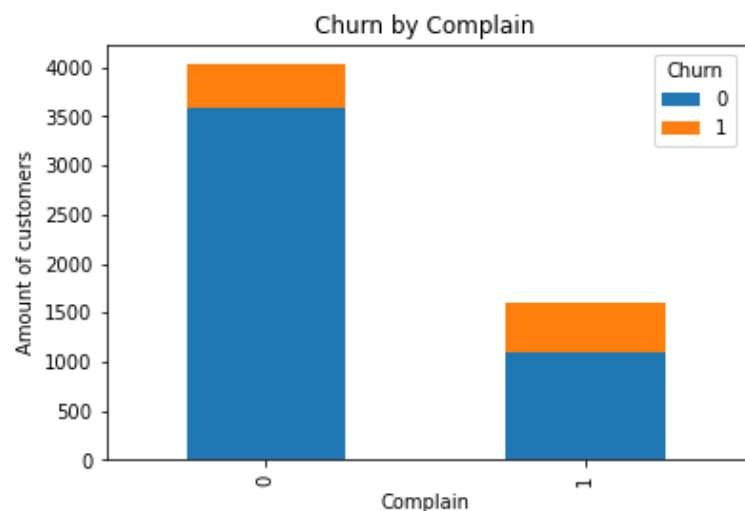


Figure 24

3.7. Hyperparameter tuning

The model with the best parameters (XGBoost) was then tuned for recall with RandomizedSearchCV. The parameters used are in Table 13 and the parameters chosen for the best model are presented in Table 14. For the comparison, there are also the parameters from when the model is tuned for F1 score instead of recall.

Table 14. The parameters used for tuning

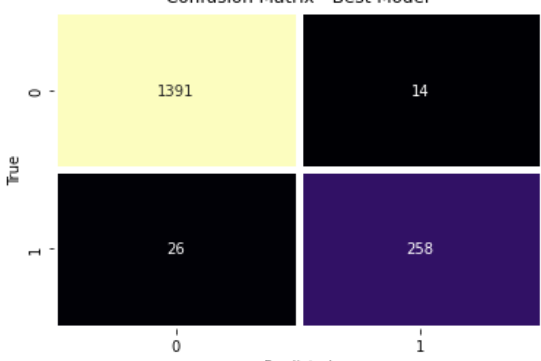
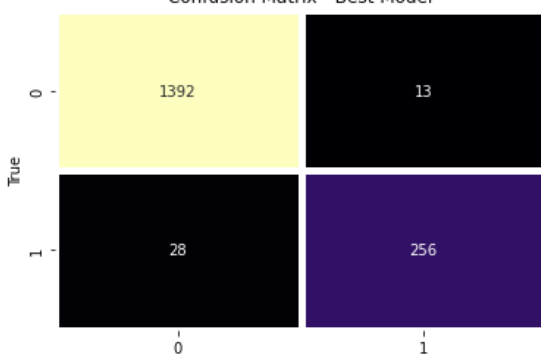
<pre>hyperparameters = {'n_estimators': stats.randint(150, 1000), 'learning_rate': stats.uniform(0.01, 0.59), 'subsample': stats.uniform(0.3, 0.6), 'max_depth': [3, 4, 5, 6, 7, 8, 9], 'colsample_bytree': stats.uniform(0.5, 0.4), 'min_child_weight': [1, 2, 3, 4] }</pre>

Table 15. The best parameters chosen

Hiperparameter	Tuning for recall	Tuning for F1 score
'colsample_bytree':	0.7448613194294949	0.8659336279612143
'learning_rate':	0.17606953123678426	0.3651082667043516
'max_depth':	8	8
'min_child_weight':	1	1
'n_estimators':	725	198
'subsample':	0.5028893096373682	0.8698948331362863

The confusion matrix after tuning is presented in Table 15. When tuning for recall, accuracy peaked for 0.98 and F1 for 0.93 (for F1 score analogical – accuracy 0.98 and F1 score 0.91).

Table 16. Confusion matrix after tuning

<p>Confusion Matrix - Best Model</p>  <p>Tuning for recall</p>	<p>Confusion Matrix - Best Model</p>  <p>Tuning for F1</p>
---	--

Summary

The subject of this thesis was churn phenomenon. The churn is important issue for E-commerce companies. 82% of companies claim that retention is cheaper than acquisition and even a small 2% increase in retention can lower costs by as much as 10% (<https://www.superoffice.com/blog/reduce-customer-churn/>, 2022).

The aim of this thesis was to analyse the data about the churn from E-commerce company. The objectives were to check which characteristics the churning customers have, and which model would be the best to analyse the churn phenomenon. The research aim and objectives were achieved.

The classification methods used for prediction of churn were Decision Tree, Bagging, Random Forest, Logistic Regression, Linear Discriminant Analysis, XGBoost, CatBoost and Neural Network. The best algorithms were XGBoost (accuracy 0.97, F1 score 0.89) and Bagging (accuracy 0.96, F1 score 0.87). After hyperparameter tuning XGBoost for recall the accuracy skyrocketed to 0.98 and F1 score to 0.93.

The churn rate in the company is 17% which is high and should be reduced. The most important features for XGBoost are Tenure and Complain. From the business side, it would be recommended to take care about them as a priority. The new customers are the first one to churn. I advise to provide an incentive for customers with low tenure in organisation. One way to approach that could be a giveaway of discount coupons upon the first order.

The special care should be taken when handling complaints. More than a third of customers made a complaint only last month. It is a huge amount so it would be thoughtful to analyse these complaints to find out what is a main issue and to work on it. The customer service should be improved as well. After reducing the complaint rate, the churn rate should fall.

The churn issue in this E-commerce company can be explored much further. Next step can be customer segmentation and different strategies can be implemented for different groups. For example, the most valuable customers should be taken extra care of because they bring in the biggest revenue.

References

- <https://builtin.com/data-science/random-forest-algorithm>. (2022, August 18).
- <https://catboost.ai/>. (2022, July 31).
- <https://dictionary.cambridge.org/pl/dictionary/english/churn> . (2022, August 03).
- <https://machinelearningmastery.com/bagging-ensemble-with-python/>. (2022, August 18).
- <https://matplotlib.org/>. (2022, July 31).
- <https://pandas.pydata.org/>. (2022, July 31).
- <https://seaborn.pydata.org/introduction.html>. (2022, July 31).
- <https://towardsdatascience.com/accuracy-recall-precision-f-score-specificity-which-to-optimize-on-867d3f11124>. (2022, August 03).
- <https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052>. (2022, August 18).
- <https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc>. (2022, August 19).
- <https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5>. (2022, August 26).
- <https://www.kaggle.com/datasets/ankitverma2010/ecommerce-customer-churn-analysis-and-prediction>. (2022, July 31).
- <https://www.kdnuggets.com/2020/12/xgboost-what-when.html>. (2022, August 18).
- <https://www.mltut.com/linear-discriminant-analysis-python-complete-and-easy-guide/>. (2022, August 19).
- <https://www.paddle.com/blog/churn-analysis>. (2022, July 31).
- <https://www.superoffice.com/blog/reduce-customer-churn/>. (2022, September 02).
- https://www.tutorialspoint.com/scikit_learn/scikit_learn_introduction.htm. (2022, July 31).
- https://www.w3schools.com/python/numpy/numpy_intro.asp. (2022, July 31).
- <https://www.winback.chat/blog/how-to-reduce-churn-in-ecommerce>. (2022, August 2).
- <https://xgboost.readthedocs.io/en/stable/>. (2022, July 31).
- Provost, F. i Fawcett, T. (2019). Analiza danych w biznesie. Sztuka podejmowania skutecznych decyzji. Helion.
- Raschka, S. i Mirjalili, V. (2019). Python. Uczenie maszynowe. Helion.

Appendix 1 – Code

Table 17. Libraries

```
import uuid

from matplotlib import pyplot as plt

from sklearn import datasets

from sklearn.tree import DecisionTreeClassifier

from sklearn.tree import export_text

from sklearn import tree

from sklearn.model_selection import train_test_split, StratifiedKFold, cross_val_score

from sklearn.neural_network import MLPClassifier

from sklearn import metrics

from scipy import stats

from sklearn.pipeline import Pipeline

from sklearn.metrics import f1_score, accuracy_score, precision_score, recall_score,
confusion_matrix

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

from sklearn.ensemble import BaggingClassifier

from sklearn.feature_selection import RFE

from sklearn.pipeline import make_pipeline

from sklearn.preprocessing import StandardScaler

from fancyimpute import IterativeImputer

from skopt import BayesSearchCV
```

```
import xgboost as xgb

from xgboost import XGBClassifier

from catboost import CatBoostClassifier

from statistics import mean

from statistics import median

from dtreeviz.trees import dtreeviz

import graphviz

import sklearn

import pandas as pd

from pandas import DataFrame

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

from sklearn.linear_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.experimental import enable_iterative_imputer

from sklearn.impute import IterativeImputer

from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
```


Table 18. Countplot

```
ch = sns.countplot(x='PreferredPaymentMode', data=df)

plt.title("Preferred payment method of customer")

plt.show()
```

Table 19. Displot

```
sns.displot(x='Tenure', data=df, kind='kde')

plt.title("Distribution of tenure of customer in organization")

plt.show()
```

Table 20. Barh plot

```
plt.barh(df['NumberOfDeviceRegistered'].unique(),
df['NumberOfDeviceRegistered'].value_counts())

plt.title("Total number of deceives registered on particular customer")

plt.show()
```

Table 21. Correlation map

```
corr = df.corr()

f, ax = plt.subplots(figsize=(12, 10))

mask = np.triu(np.ones_like(corr, dtype=bool))

cmap = sns.diverging_palette(230, 20, as_cmap=True)

sns.heatmap(corr, annot=True, mask = mask, cmap=cmap)
```

Table 22. Boxplot

```
plt.figure(figsize=(20,20))

sns.boxplot(data=df, orient="h", dodge=bool, width=0.8)

plt.title('The boxplot to study outliers', fontsize =50)

plt.xlabel('Values', fontsize =30)

plt.ylabel('Variables that predict the customer churn', fontsize =30)
```

Table 23. Data split

```
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size = .3, random_state
= 42)
```

Table 24. Modelling

```
# Create Decision Tree classifier object

clf = DecisionTreeClassifier()

# Train Decision Tree Classifier

clf = clf.fit(X_train,y_train)

#Predict the response for test dataset

y_pred = clf.predict(X_test)

# Create Bagging classifier object

clf_bagging = BaggingClassifier()

# Train Bagging Classifier

clf_bagging = clf_bagging.fit(X_train,y_train)

#Predict the response for test dataset

y_pred_bagging = clf_bagging.predict(X_test)
```

```
# Create CatBoost classifier object

clf_catboost = CatBoostClassifier()

# Train CatBoost Classifier

clf_catboost = clf_catboost.fit(X_train,y_train)

#Predict the response for test dataset

y_pred_catboost = clf_catboost.predict(X_test)
```

```
# Create Neural Network object

clf_nn = MLPClassifier(max_iter=2000, random_state=1)

# Train Neural Network

clf_nn = clf_nn.fit(X_train,y_train)

#Predict the response for test dataset

y_pred_nn = clf_nn.predict(X_test)
```

```
# Create Random Forest classifier object

clf_randomforest = RandomForestClassifier()

# Train CatBoost Classifier

clf_randomforest = clf_randomforest.fit(X_train,y_train)

#Predict the response for test dataset

y_pred_randomforest = clf_randomforest.predict(X_test)
```

```
# Create Logistic Regression classifier object

clf_lg = LogisticRegression(solver='lbfgs', max_iter=2000)

# Train LogisticRegression Classifier

clf_lg = clf_lg.fit(X_train,y_train)
```

```
#Predict the response for test dataset
```

```
y_pred_lg = clf_lg.predict(X_test)
```

```
# Create Linear Discriminant Analysis classifier object
```

```
clf_lda = LinearDiscriminantAnalysis()
```

```
# Train LinearDiscriminantAnalysis Classifier
```

```
clf_lda = clf_lda.fit(X_train,y_train)
```

```
#Predict the response for test dataset
```

```
y_pred_lda = clf_lda.predict(X_test)
```

```
# Create XGBoost classifier object
```

```
clf_xgb = XGBClassifier()
```

```
# Train XGBoost Classifier
```

```
clf_xgb = clf_xgb.fit(X_train,y_train)
```

```
#Predict the response for test dataset
```

```
y_pred_xgb = clf_xgb.predict(X_test)
```

Table 25. Classification reports

<code>print(metrics.classification_report(y_test, y_pred))</code>
<code>print(metrics.classification_report(y_test, y_pred_bagging))</code>
<code>print(metrics.classification_report(y_test, y_pred_catboost))</code>
<code>print(metrics.classification_report(y_test, y_pred_nn))</code>
<code>print(metrics.classification_report(y_test, y_pred_randomforest))</code>
<code>print(metrics.classification_report(y_test, y_pred_lg))</code>
<code>print(metrics.classification_report(y_test, y_pred_lda))</code>
<code>print(metrics.classification_report(y_test, y_pred_xgb))</code>

Table 26. Confusion matrixes

<pre>sns.heatmap(confusion_matrix(y_test,y_pred),annot=True,linewidths=5,cbar=False, fmt="g", cmap="magma") plt.title('Confusion Matrix - Decision Tree') plt.ylabel('True') plt.xlabel('Predicted') plt.show()</pre>
<pre>sns.heatmap(confusion_matrix(y_test,y_pred_bagging),annot=True,linewidths=5,cbar=False, fmt="g", cmap="magma") plt.title('Confusion Matrix - Bagging') plt.ylabel('True') plt.xlabel('Predicted') plt.show()</pre>

```
sns.heatmap(confusion_matrix(y_test,y_pred_catboost),annot=True,linewidths=5,cbar=False,
fmt="g", cmap="magma")
```

```
plt.title('Confusion Matrix - CatBoost')
```

```
plt.ylabel('True')
```

```
plt.xlabel('Predicted')
```

```
plt.show()
```

```
sns.heatmap(confusion_matrix(y_test,y_pred_nn),annot=True,linewidths=5,cbar=False,
fmt="g", cmap="magma")
```

```
plt.title('Confusion Matrix - Neural Network')
```

```
plt.ylabel('True')
```

```
plt.xlabel('Predicted')
```

```
plt.show()
```

```
sns.heatmap(confusion_matrix(y_test,y_pred_randomforest),annot=True,linewidths=5,cbar=False,
fmt="g", cmap="magma")
```

```
plt.title('Confusion Matrix - Random Forest')
```

```
plt.ylabel('True')
```

```
plt.xlabel('Predicted')
```

```
plt.show()
```

```
sns.heatmap(confusion_matrix(y_test,y_pred_lg),annot=True,linewidths=5,cbar=False,
fmt="g", cmap="magma")
```

```
plt.title('Confusion Matrix - Logistic regression')
```

```
plt.ylabel('True')
```

```
plt.xlabel('Predicted')
```

```
plt.show()
```

```
sns.heatmap(confusion_matrix(y_test,y_pred_lda),annot=True,linewidths=5,cbar=False,
fmt="g", cmap="magma")
```

```
plt.title('Confusion Matrix - Linear Discriminant Analysis')
```

```
plt.ylabel('True')
```

```
plt.xlabel('Predicted')
```

```
plt.show()
```

```
sns.heatmap(confusion_matrix(y_test,y_pred_xgb),annot=True,linewidths=5,cbar=False,
fmt="g", cmap="magma")
```

```
plt.title('Confusion Matrix - XGBoost')
```

```
plt.ylabel('True')
```

```
plt.xlabel('Predicted')
```

```
plt.show()
```

Table 27. Importances

```
#extract importance
```

```
importance = pd.DataFrame({'feature': X_train.columns, 'importance' :
np.round(clf.feature_importances_, 3)})
```

```
importance.sort_values('importance', ascending=False, inplace = True)
```

```
print(importance)
```

```
#extract importance
```

```
importance = pd.DataFrame({'feature': X_train.columns, 'importance' :
np.round(clf_randomforest.feature_importances_, 3)})
```

```
importance.sort_values('importance', ascending=False, inplace = True)
```

```
print(importance)
```

```
#extract importance

importance = pd.DataFrame({'feature': X_train.columns, 'importance' :
np.round(clf_catboost.feature_importances_, 3)})

importance.sort_values('importance', ascending=False, inplace = True)

print(importance)
```

```
#extract importance

importance = pd.DataFrame({'feature': X_train.columns, 'importance' :
np.round(clf_xgb.feature_importances_, 3)})

importance.sort_values('importance', ascending=False, inplace = True)

print(importance)
```

Table 28. Hyperparameter tuning

```
#tuning for F1

search2=RandomizedSearchCV(clf_xgb,hyperparameters, n_iter=100, n_jobs=-1, cv=skfcv,
scoring='f1')

search2.fit(X_train, y_train)

print(search2.best_score_)

print(search2.best_params_)

#use the best model parameters to predict the test sample and print the results

best_model2=search2.best_estimator_

print(f1_score(y_test,best_model2.predict(X_test)))

sns.heatmap(confusion_matrix(y_test, best_model2.predict(X_test)), annot=True,
linewidths=5,cbar=False, fmt="g", cmap="magma")

plt.title('Confusion Matrix - Best Model')
```



```
plt.ylabel('True')
```

```
plt.xlabel('Predicted')
```

```
plt.show()
```

```
#tuning for recall
```

```
search4=RandomizedSearchCV(clf_xgb, hyperparameters, n_iter=100, cv=skfcv,  
scoring='recall')
```

```
search4.fit(X_train, y_train)
```

```
print(search4.best_score_)
```

```
print(search4.best_params_)
```

```
best_model4=search4.best_estimator_
```

```
print(recall_score(y_test,best_model4.predict(X_test)))
```

```
sns.heatmap(confusion_matrix(y_test, best_model4.predict(X_test)), annot=True,  
linewidths=5,cbar=False, fmt="g", cmap="magma")
```

```
plt.title('Confusion Matrix - Best Model')
```

```
plt.ylabel('True')
```

```
plt.xlabel('Predicted')
```

```
plt.show()
```

Appendix 2 – Decision Tree

Visualization with plot_tree

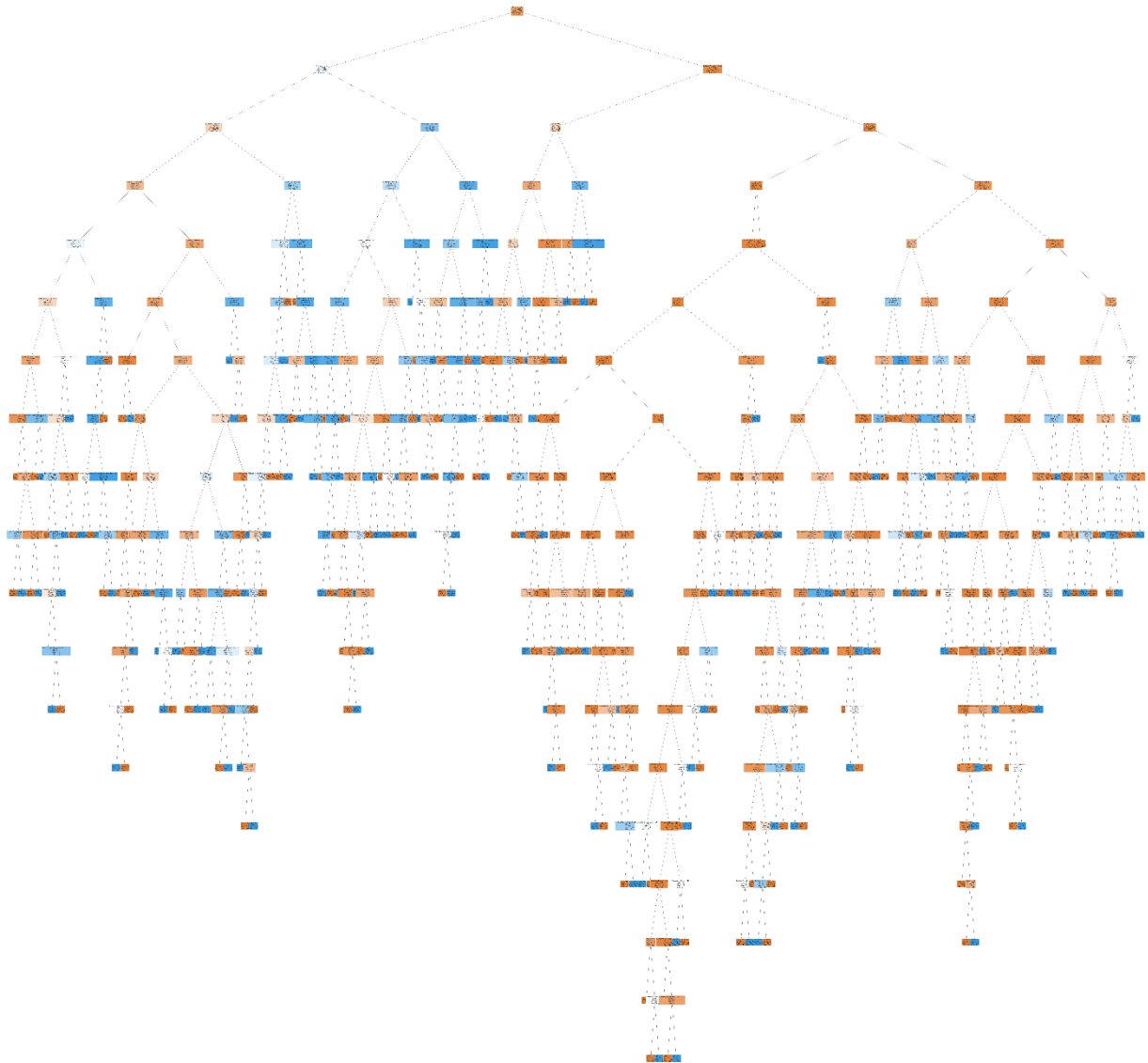


Figure 25

Visualization with dtreeviz (a piece)

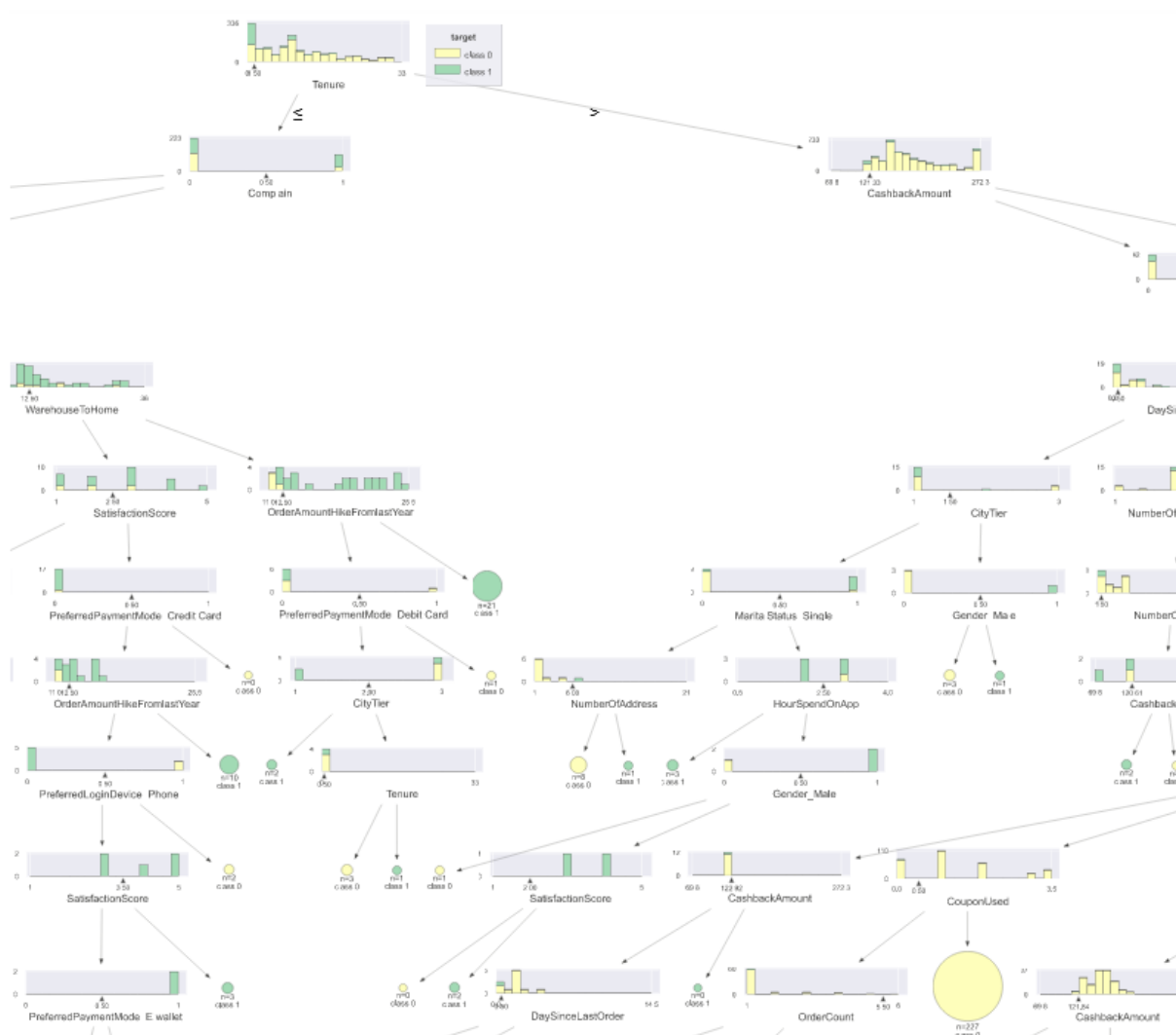


Figure 26

Visualization with graphviz (a piece)

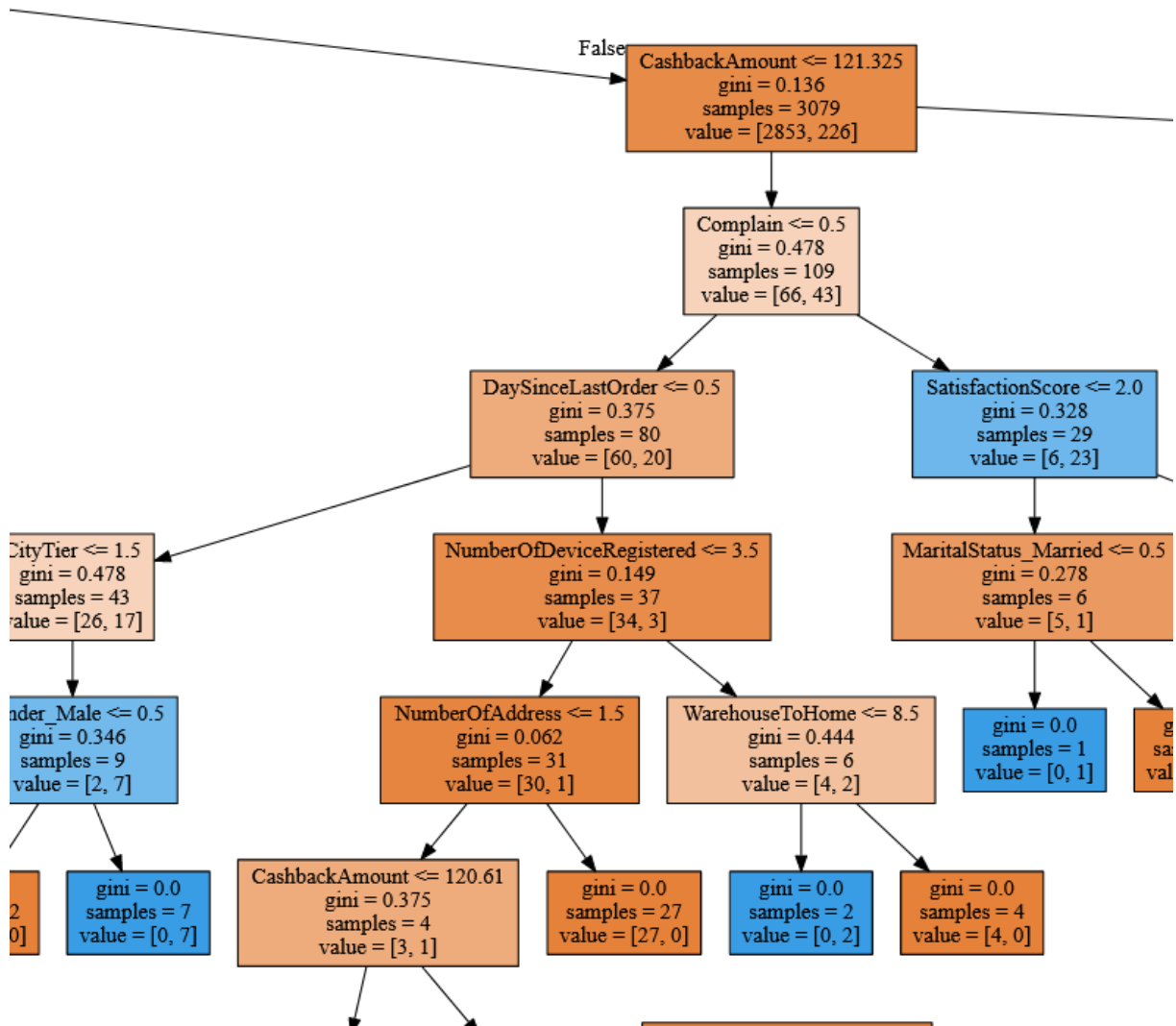


Figure 27